自定义view、动画、Material Design

1. 自定义View
   1. 自定义View的属性

自定义View的属性，首先在res/values/  下建立一个attrs.xml ， 在里面定义我们的属性和声明我们的整个样式。

1. **<?xml** version="1.0" encoding="utf-8"**?>**
2. **<resources>**
4. **<attr** name="titleText" format="string" **/>**
5. **<attr** name="titleTextColor" format="color" **/>**
6. **<attr** name="titleTextSize" format="dimension" **/>**
8. **<declare-styleable** name="CustomTitleView"**>**
9. **<attr** name="titleText" **/>**
10. **<attr** name="titleTextColor" **/>**
11. **<attr** name="titleTextSize" **/>**
12. **</declare-styleable>**
14. **</resources>**
    1. 在View的构造方法中获得我们自定义的属性
15. /\*\*
16. \* 文本
17. \*/
18. **private** String mTitleText;
19. /\*\*
20. \* 文本的颜色
21. \*/
22. **private** **int** mTitleTextColor;
23. /\*\*
24. \* 文本的大小
25. \*/
26. **private** **int** mTitleTextSize;
28. /\*\*
29. \* 绘制时控制文本绘制的范围
30. \*/
31. **private** Rect mBound;
32. **private** Paint mPaint;
34. **public** CustomTitleView(Context context, AttributeSet attrs)
35. {
36. **this**(context, attrs, 0);
37. }
39. **public** CustomTitleView(Context context)
40. {
41. **this**(context, **null**);
42. }
44. /\*\*
45. \* 获得我自定义的样式属性
46. \*
47. \* @param context
48. \* @param attrs
49. \* @param defStyle
50. \*/
51. **public** CustomTitleView(Context context, AttributeSet attrs, **int** defStyle)
52. {
53. **super**(context, attrs, defStyle);
54. /\*\*
55. \* 获得我们所定义的自定义样式属性
56. \*/
57. TypedArray a = context.getTheme().obtainStyledAttributes(attrs, R.styleable.CustomTitleView, defStyle, 0);
58. **int** n = a.getIndexCount();
59. **for** (**int** i = 0; i < n; i++)
60. {
61. **int** attr = a.getIndex(i);
62. **switch** (attr)
63. {
64. **case** R.styleable.CustomTitleView\_titleText:
65. mTitleText = a.getString(attr);
66. **break**;
67. **case** R.styleable.CustomTitleView\_titleTextColor:
68. // 默认颜色设置为黑色
69. mTitleTextColor = a.getColor(attr, Color.BLACK);
70. **break**;
71. **case** R.styleable.CustomTitleView\_titleTextSize:
72. // 默认设置为16sp，TypeValue也可以把sp转化为px
73. mTitleTextSize = a.getDimensionPixelSize(attr, (**int**) TypedValue.applyDimension(
74. TypedValue.COMPLEX\_UNIT\_SP, 16, getResources().getDisplayMetrics()));
75. **break**;
77. }
79. }
80. a.recycle();
82. /\*\*
83. \* 获得绘制文本的宽和高
84. \*/
85. mPaint = **new** Paint();
86. mPaint.setTextSize(mTitleTextSize);
87. // mPaint.setColor(mTitleTextColor);
88. mBound = **new** Rect();
89. mPaint.getTextBounds(mTitleText, 0, mTitleText.length(), mBound);
91. }
    * 1. 重写onMesure
92. @Override
93. **protected** **void** onMeasure(**int** widthMeasureSpec, **int** heightMeasureSpec)
94. {
95. **int** widthMode = MeasureSpec.getMode(widthMeasureSpec);
96. **int** widthSize = MeasureSpec.getSize(widthMeasureSpec);
97. **int** heightMode = MeasureSpec.getMode(heightMeasureSpec);
98. **int** heightSize = MeasureSpec.getSize(heightMeasureSpec);
99. **int** width;
100. **int** height ;
101. **if** (widthMode == MeasureSpec.EXACTLY)
102. {
103. width = widthSize;
104. } **else**
105. {
106. mPaint.setTextSize(mTitleTextSize);
107. mPaint.getTextBounds(mTitle, 0, mTitle.length(), mBounds);
108. **float** textWidth = mBounds.width();
109. **int** desired = (**int**) (getPaddingLeft() + textWidth + getPaddingRight());
110. width = desired;
111. }
113. **if** (heightMode == MeasureSpec.EXACTLY)
114. {
115. height = heightSize;
116. } **else**
117. {
118. mPaint.setTextSize(mTitleTextSize);
119. mPaint.getTextBounds(mTitle, 0, mTitle.length(), mBounds);
120. **float** textHeight = mBounds.height();
121. **int** desired = (**int**) (getPaddingTop() + textHeight + getPaddingBottom());
122. height = desired;
123. }


127. setMeasuredDimension(width, height);
128. }
     1. 重写onDraw
129. @Override
130. **protected** **void** onDraw(Canvas canvas)
131. {
132. mPaint.setColor(Color.YELLOW);
133. canvas.drawRect(0, 0, getMeasuredWidth(), getMeasuredHeight(), mPaint);
135. mPaint.setColor(mTitleTextColor);
136. canvas.drawText(mTitleText, getWidth() / 2 - mBound.width() / 2, getHeight() / 2 + mBound.height() / 2, mPaint);
137. }

二、动画

(1) Tween Animation (变换动画)

　　Tween Animation按效果可以分为四种:Alpha,Scale,Translate,Rotate,它们有一下共同属性:

　　1. Duration: 动画持续时间(毫秒);

　　2. fillAfter: 设置为True,动画转换在动画结束后被应用;

　　3. fillBefore: 设置为True,动画转换在动画开始前被应用;

　　4. interpolator: 动画插入器(加速,减速插入器);

　　5. repeatCount: 动画重复次数;

　　6. repeatMode: 正序重复/倒序重复

　　7. startOffSet: 动画之间的时间间隔

Ⅰ静态实现

1.Alpha (渐变透明度动画)

MainActivity.java主要代码:

ImageView image = (ImageView) findViewById(id.image);

Animation loadAnimation = AnimationUtils.loadAnimation(this, R.anim.alpha);

image.startAnimation(loadAnimation);

alpha.xml

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android" >

<alpha

android:duration="1000"

android:fromAlpha="0.1"

android:toAlpha="1.0" >

</alpha>

</set>

[复制代码](javascript:void(0);)

2.Scale (渐变尺寸缩放动画)

ImageView image = (ImageView) findViewById(id.image);

Animation loadAnimation = AnimationUtils.loadAnimation(this, R.anim.scale);

image.startAnimation(loadAnimation);

scale.xml

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android" >

<scale

android:duration="2000"

android:fillAfter="false"

android:fromXScale="0.0"

android:fromYScale="0.0"

android:interpolator="@android:anim/accelerate\_decelerate\_interpolator"

android:pivotX="50%"

android:pivotY="50%"

android:toXScale="1.0"

android:toYScale="1.0" />

</set>

[复制代码](javascript:void(0);)

3.Translate (位置移动动画)

ImageView image = (ImageView) findViewById(id.image);

Animation loadAnimationloadAnimation = AnimationUtils.loadAnimation(this, R.anim.translate);

image.startAnimation(loadAnimation);

translate.xml

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android" >

<translate

android:duration="1000"

android:fromXDelta="10"

android:fromYDelta="10"

android:toXDelta="100"

android:toYDelta="100" />

</set>

[复制代码](javascript:void(0);)

4.Rotate (旋转动画)

ImageView image = (ImageView) findViewById(id.image);

Animation loadAnimation loadAnimation = AnimationUtils.loadAnimation(this, R.anim.rotate);

image.startAnimation(loadAnimation);

rotate.xml

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android" >

<rotate

android:duration="1000"

android:fromDegrees="0"

android:interpolator="@android:anim/accelerate\_decelerate\_interpolator"

android:pivotX="50%"

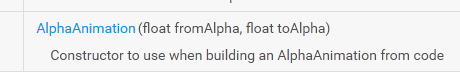
android:pivotY="50%"

android:toDegrees="+360" />

</set>

Ⅱ动态实现

1.AlphaAnimation（透明度）

   
参数：   
fromAlpha：开始透明度（数值在0-1之间）   
toAlpha：结束透明度（数值在0-1之间）

//创建AlphaAnimation（透明度动画）

AlphaAnimation animation=new AlphaAnimation(0.0f, 1.0f);

//设置动画时间

animation.setDuration(2000);

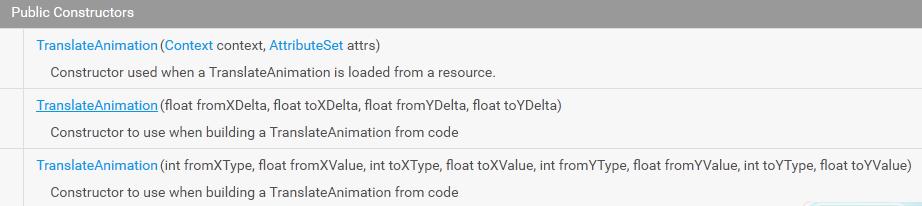
//设置动画重复次数

animation.setRepeatCount(2);

//开启动画效果

imageview.startAnimation(animation);

## 2.TranslateAnimation（平移）

 参数：   
fromXDela/toXDela：开始/结束的x坐标   
fromYDela/toYDela：开始/结束的y坐标

//创建TranslateAnimation位移动画

TranslateAnimation trananimation=new TranslateAnimation(-imageview.getMeasuredWidth(), 0,0,300);

//设置动画时间

trananimation.setDuration(2000);

//设置是否记录移动后的位置，true时动画将停留在当前位置，false将回到开始位置

trananimation.setFillAfter(true);

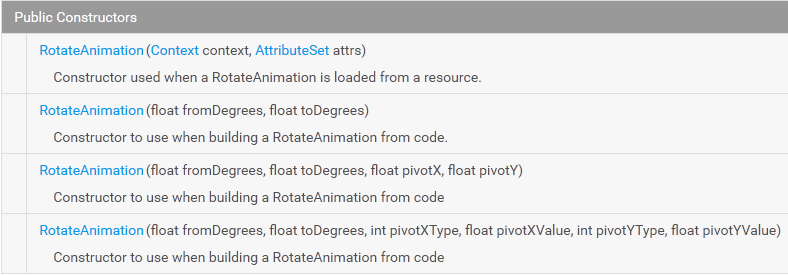
//设置插值器，可以理解为用于改变运动形式的东西

//（现在设置的运动形式类似于自由落体，会有弹跳效果）

trananimation.setInterpolator(new BounceInterpolator());

imageview.startAnimation(trananimation);

## 3.RotateAnimation(旋转)

   
参数：   
fromDegrees/toDegrees:开始/结束角度   
pivotX/pivotY:分别为旋转动画相对于x，y的坐标开始位置（相对值）   
注：区别与X，Y：是绝对坐标

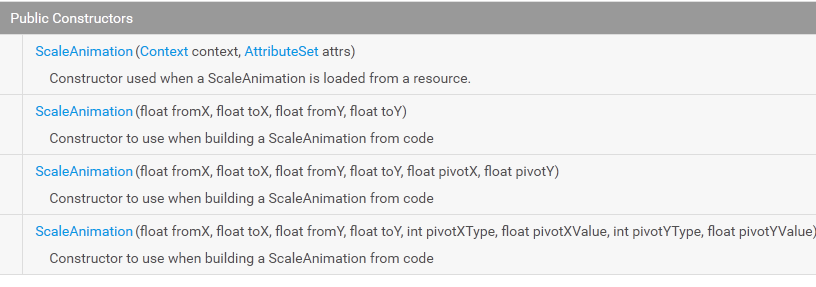
//参数：从0度旋转100度，旋转中心坐标（100，100）

RotateAnimation rotateAnimation=new RotateAnimation(0, 180, 100, 100);

rotateAnimation.setDuration(2000);

imageview.startAnimation(rotateAnimation);

## 4.ScaleAnimation（放缩）



//参数：X轴从0.2倍放大到1倍（原图）Y轴也一样

ScaleAnimation scaleanimation=new ScaleAnimation(0.2f, 1f, 0.2f, 1f);

scaleanimation.setDuration(2000);

imageview.startAnimation(scaleanimation);

## 5.动画监听（setAnimationListener）

对位移动画添加动画监听器，动画完成就开始第二个动画

TranslateAnimation trananimation2=new TranslateAnimation(-imageview.getMeasuredWidth(), 0,0,100);

trananimation2.setDuration(2000);

//设置是否记录移动后的位置，true时动画将停留在原来位置

trananimation2.setFillAfter(true);

imageview.startAnimation(trananimation2);

//监听动画trananimation2，当它运行完时开始第二个动画AlphaAnimation

trananimation2.setAnimationListener(new AnimationListener() {

//监听动画开始时弹出TOAST

@Override

public void onAnimationStart(Animation animation) {

Toast.makeText(getApplicationContext(), "动画展示开始", Toast.LENGTH\_SHORT).show();

}

@Override

public void onAnimationRepeat(Animation animation) {

} //动画结束时开启第二个

@Override

public void onAnimationEnd(Animation animation) {

AlphaAnimation animation2=new AlphaAnimation(0.0f, 1.0f);

animation2.setDuration(2000);

animation2.setRepeatCount(2);

imageview.startAnimation(animation2);

}

});

## 6.AnimationSet动画组合

1、创建AnimationSet 　　 2、添加动画

//创建几个动画效果

AlphaAnimation animation3=new AlphaAnimation(0.0f, 1.0f);

animation3.setDuration(2000);

animation3.setRepeatCount(2);

//创建AnimationSet 动画组合

AnimationSet set=new AnimationSet(true);

TranslateAnimation trananimation3=new TranslateAnimation(-imageview.getMeasuredWidth(), 0,0,100);

trananimation3.setDuration(2000);

RotateAnimation rotateAnimation2=new RotateAnimation(0, 360, 200, 200);

rotateAnimation2.setDuration(2000);

//添加组合动画

set.addAnimation(animation3);

set.addAnimation(trananimation3);

set.addAnimation(rotateAnimation2);

imageview.startAnimation(set);